# History of JAVA: -

* In **1990 Sun micro system** try to build set up box. They assign to Green Team. **James Gosling** is the team leader of that team is also known as the **father of the Java.**
* Earlier name of Java is known as OAK(OAK Tree represent to strength)

Facts of OAK-

* It is the National Tree of USA, Italy and Numbers of country etc.
* A company named OAK technology is already present in USA. So the founder of OAK language change its name to JAVA( which represent the location java island in Indonesia where, coffee 1st discovered)

# Version hierarchy of JAVA:

1. JDK 1.0------->1995
2. JDK 1.2------->1997

All are the Stable Version of Java

1. JDK 1.5------->2004
2. JDK 1.8------->2014

Here we are going to use JDK 1.8

1. JDK ……………
2. JDK 15-------->2020
3. JDK 17-------->2021

# What is JAVA: -

* Java is a general-purpose programming language.
* General purpose means it is used in both hardware and software Industries.
* But JAVA is very much famous in software industry to develop the application.

# Features of JAVA: -

1. It is simple.
2. Secure
3. Platform Independent
4. Portable
5. Robust
6. Object oriented programming language
7. Class structure programming language
8. Type safe programming language
9. Interpreted programming language
10. WORA language (Write once run anywhere)

# Type of Application: -

1. The application which can run without internet called 🡪 Web Application.
2. The application which can run without internet called 🡪 Standalone Application.

# Platform of JAVA: -

There are 3 types of platforms in Java. -

# JDK Architecture-

execution (JRE)

Coding

Pre define syntax library

JVM

Compilation

x.class

Byte code (0,1)

X.Java

X.Java

**JRE**

**JDK** code (0,1)

**JDK**-

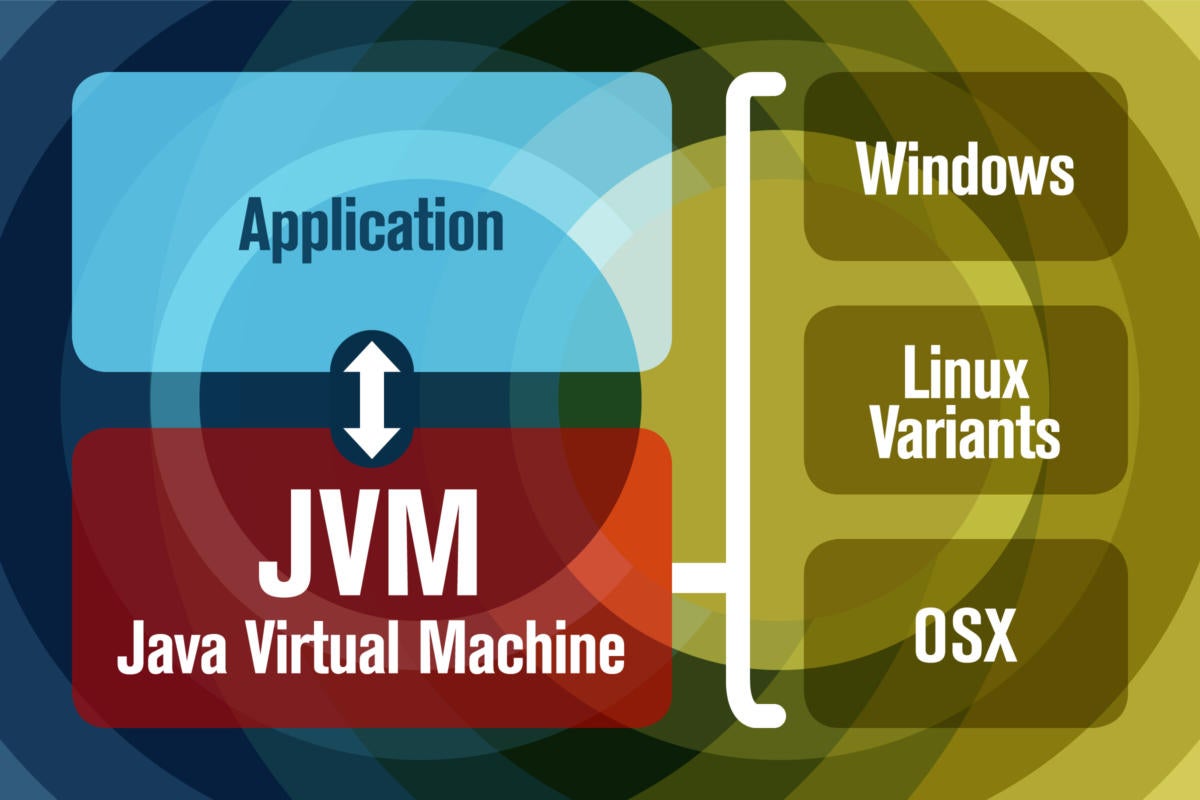
If **jdk compilation is fail** then jdk will show **Fully qualified Information will show**.

1. **Where is the error?**
2. **What is the error?**
3. **How is the error?**
4. **How many error?**

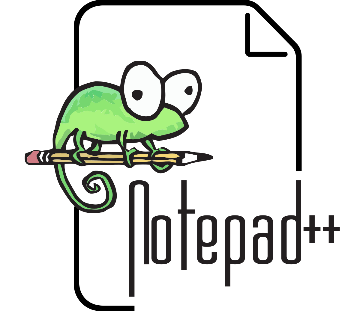
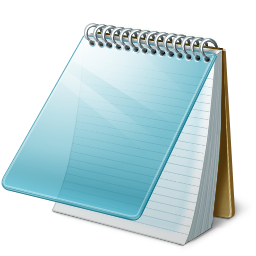
* It stands for java development kit.
* It provides the environment for coding & compilation.

**JRE**-

* It stands java run time environment.
* It provides the environment for execution.

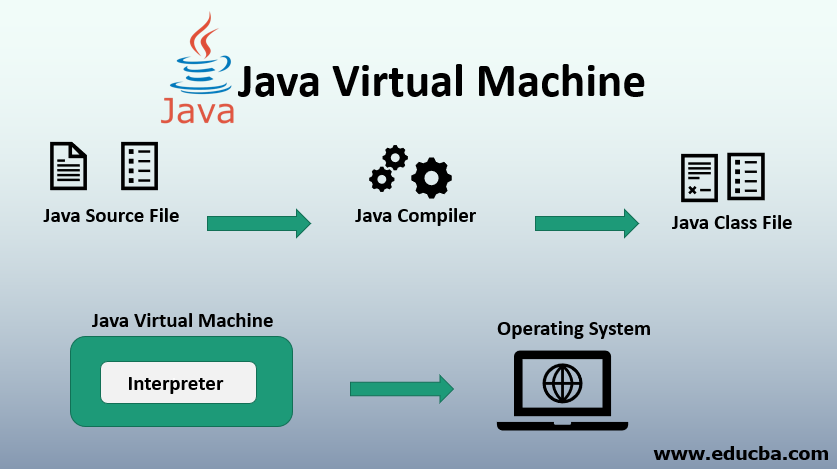
**JVM**-

* It stands for java virtual machine.
* JRE is just a folder name inside that a powerful machine is present that is JVM.
* That JVM is fully responsible for the execution of Java Program.

CODING –

* It is done by programmer.
* Usually programmer will write the code in editors.

Ex- notepad, notepad++, edit+

* Usually programmer done with the program save the file with ‘file\_name.java’ extension.

Compilation –

* It is done by compiler.
* It is the process of cross-checking ‘**file\_name.java**’ file with predefine syntax library to encounter the syntax level mistake.

## Error Case-

1. Compilation pass
2. Compilation fail

**Compilation pass-**

* If compilation pass than ‘**file\_name.java**’ file convert to ‘**class\_name.class**’ file contain ‘0’ and ‘1’. This is known as machine level language.
* Computer can understand only machine level language. Because of this ‘**class\_name.class**’ file become portable, platform independent, secure & WORA language.

**Compilation Fail-**

* If the compilation become fail then ‘**file\_name.java**’ file convert to ‘**class\_name.class**’ and we will get a compile time error.
* This compile time error contain Fully Qualified Information about the error(**FQI**).

**FQI-** FQI means we will know the following question answer.

* **What is the error?**
* **Where is the error?**
* **How many error?**
* **How to resolve the error?**

**Example**-

Class Test

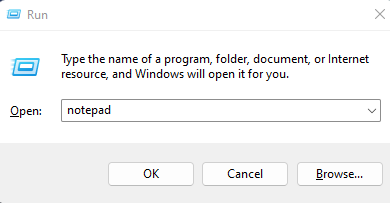
{

Public static void main(string[] args)

{

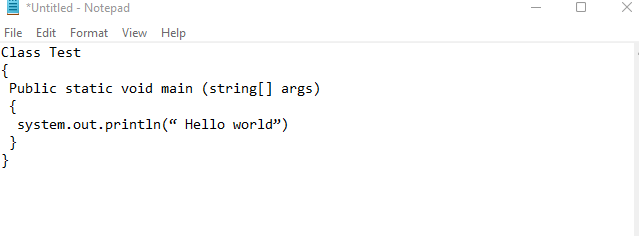
system.out.println(“ Hello world”);

}

}

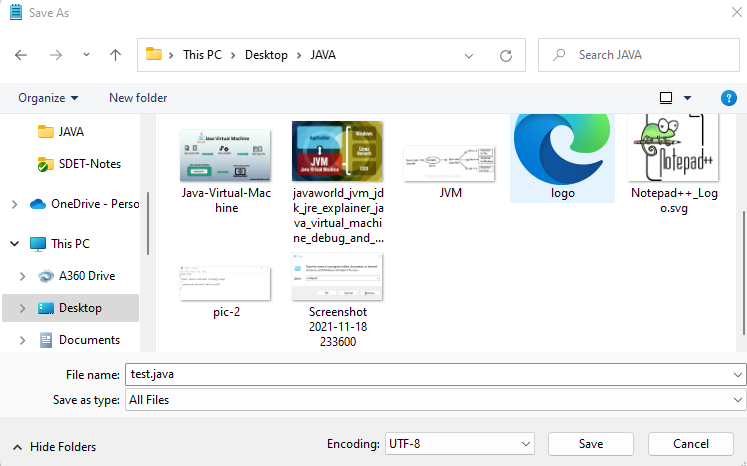
**For execution Process**-

* To open notepad

1. Win+R
2. Enter
3. Type ‘notepad’ in run

* To save the file-

1. File
2. Got to Save option
3. Save file\_name as ‘filename.java’
4. Change to all files, then click the save option.



**File\_name.java**

**File type ‘All type**

The **compilation and execution** takes place in system console or **command promt.**

**To open command promt and execution process –**

1. Win+R->
2. Type ‘cmd’
3. Ok/Enter

**Condtion: -**

1. **With in same Driver**

(cd- change directory, here from D to D/ODM18

Cd folder path

Example =

1. **With in same Driver**

(cd- change directory, here from D to D/ODM18

Cd folder path

Example =

* **Code for execution: -**

**Syntax- javac file\_name.java**

* **Code for execution: -**

**Syntax- java class\_name**

**Case-1->**

**Program:-**

class Test

{

public static void main(String [ ] args)

{

system.out.println(“ Hello world”);

}

}

**Code for compilation-**

* **Ex- javac sample.java**

**Folder Name**

**Code for execution-**

* **Ex- java Demo**

**Variable & methods**

**Note:-**

* **To avoid the future execution according to java coding standard always proving same file name & variable & methods.**

**Case-2->**

**Program:-**

**Compilation – Pass ✔**

**Execution – fail ❌**

Class A

{

}

**Error- mainmethod not found in class A**

**Note:-**

* **When we get error at execution failed it is runtime error.**
* **Without the mainmethod we can compile a java program.**
* **If you want to execute a java program then mainmethod is compulsory.**

**Case-3->**

**Program:-**

Class A

{

}

Class B

{

}

Class C

{

}

**Note:-**

* **In a class we can write n.. numbers of class and after compilation, it will**

**Create separate class for each class.**

**Case-4->**

**Program: -**

Class A

{

}

Class A

{

}

Class A

{

}

**Note: -**

* **In a class we can write one variable & methods (Variable & methods are case-sensitive and always start with upper case).**

# Class Structure of Java-

class Demo

keyword- Always start with **lowercase**

Identifier- May be **UPPERCASE** May be **lowercase**

Keyword Identifier

{

Variables

Methods

Class Member

Blocks

Constructs

}

public static void main(String [] args)

{

system.out.println(“Hai”);

}

O/p- Hai

**Java Class-**

* Class is an area where we declare class member i.e. variables, methods, block and constructors.
* Entire java program we will write inside the class only.
* If we write outside the java class then we will get compile time error.
* To execute a java mainmethod is mandatory.

**Keywords-**

* Keywords are the important words which have the functionality in a short manner.
* In **java** there are **total 53 keywords** present.
* **All keywords** we have to write in **lower case**.

**Identifiers-**

* These are used to identify the class or class members.
* Identifiers may exist in upper case and lower case also.
* Identifiers may be predefined / user defined.

## Rules to provide the variable & methods

1. The variable & methods always start with upper case.

Example – class Demo**✔** Example – class demo **❌**

1. Numeric Alpha variable & methods not allowed.

Example – class 21demo **❌**

1. Alpha Numeric variable & methods allowed.

Example – class demo12 **✔**

1. Space is not allowed in between variable & methods.

Example – class de mo **❌**

1. Special character not allowed except ‘$, \_’ in variable & methods.

Example – class Demo\_02**✔** Example – class demo@02 **❌**

## Rules to provide the variable & methods

1. The variable & methods always start with lower case.

Example – main() **✔** Example – Main() **❌**

1. Numeric Alpha variable & methods not allowed.

Example – 1main() **❌**

1. Alpha Numeric variable & methods allowed.

Example – sum1 () **✔**

1. Space is not allowed in between variable & methods.

Example – sum one () **❌**

1. Special character not allowed except ‘$, \_’ in variable & methods.

Example – sum\_one () **✔** Example – sum@one () **❌**

**Note -**

* **Always we should start our class name then it should always start with upper case & method should in lower case.**

**Example- class Demo✔ class demo ❌**

**Method ( ) ❌ method ( )✔**

## Rule Printing Statement in java

* Printing statement are used to print the output in the console.
* There are two printing statements are in java.

**Ex-** 1.system.out.println ()

2. system.out.print ()

**System.out.print ()**

**Case-1**

* It is used to print the output in the console and keep the courser in the same line.

class Demo

{

public static void main(String [ ] args)

{

system.out.print(“hello”);

system.out.print(“world”);

}

}

O/p- helloworld

**Case-2**

* In this case of system.out.print printing statement is compulsory. Otherwise we will get compile time error.

class Demo

{

public static void main(String [ ] args)

{

system.out.print(“hello”);

system.out.print( );

}

}

O/p- compile time error

**System.out.println ()**

**Case-1**

* It is used to print the output in the console and keep the courser in the next line.

class Demo

{

public static void main(String [ ] args)

{

system.out.println(“hello”);

system.out.println(“world”);

}

}

O/p- hello

world

**Case-2**

* In this case of system.out.println printing statement is not compulsory.

class Demo

{

public static void main(String [ ] args)

{

system.out.println(“hello”);

system.out.println( );

}

}

O/p- hello

## Rules for Print Statement for number

**Case-1**

class Test

{

public static void main(String [ ] args)

{

system.out.println(“100”);

O/p- 100

(it will consider ‘1’,’0’,’0’ each as individual character

}

}

**Case-2**

class Test

{

public static void main(String [ ] args)

{

O/p- 100

(it will consider 100 as Number

system.out.println(100);

}

}

**Case-3**

class Test

{

public static void main(String [ ] args)

{

O/p- Compilation time error

system.out.println(hello);

}

}

**Case-4**

class Test

{

public static void main(String [ ] args)

{

system.out.println(“A”);

O/p- A (here “A” as string)

}

}

**Case-5**

O/p- A (here ‘A’ as charater)

class Test

{

public static void main(String [ ] args)

{

system.out.println(‘A’);

}

}

# Operator Overloading

**Case-1**

class **Test**

{

public static void main(String [ ] args)

{

O/p- Hello --> String

123 --> String

123 --> Number

System.out.println(“Hello”);

System.out.println(“123”);

System.out.println(123);

}

}

**Case-2**

class **Test**

{

**public static void** **main**(**String** [ ] args)

{

**System.**out**.println**(12+12);

O/p- 24 --> String

12+12 --> String

1212 --> Number

**System.**out**.println** (“12+12”);

**System.**out**.println** (“12”+”12”);

}

}

***\*Note-***

1. ‘+’ Sign with Numbers 🡪 Addition
2. ‘+’ Sign within String🡪 Nothing (Print as it is )
3. ‘+’ Sign with String 🡪 Concatenation🡪Joining of 2 information is called concatenation.

O/p- 121212 --> String

**Case-3**

class **Test**

{

**public static void** **main**(**String** [ ] args)

{

System.out.println(“12”+12+12);

}

**JAVA reads statement left to right (String + Number= String)**

O/p- 121212 --> String

}

1. If any one information is string it will perform throughout concatenation.

**Case-4**

class **Test**

{

**public static void** **main**(**String** [ ] args)

{

**System.**out**.println**(12+12+”12”);

}

**JAVA reads statement left to right**

O/p- 2412 --> String

}

***\*Note-***

* Whenever ‘+’ sign perform concatenation & addition at a same time then it is called as operator overloading.
* ‘+’ changes its behavior is nothing but operator overloading.

**Case-5**

class **Test**

{

**public static void** **main**(**String** [ ] args)

{ **System.**out**.println**(“Hello”+”World”);

**System.**out**.println** (“Hi”+15+30);

O/p- HelloWorld 🡪

Hi1530 🡪

45Hi 🡪

Hi45🡪

**System.**out**.println** (15+30+Hi);

**System.**out**.println** (“Hi”+(15+30));

}

}

**Case-5**

Concatenation help to give meaningful complete information

class **Test**

{

**public static void** **main**(**String** [ ] args)

O/p- 45

Addition of 15 and 20 is: 45

{

**System.**out**.println** (15+30);

**System.**out**.println** (“Addition of 15 and 20 is: “+(15+30));

}

If in the below program if we use variable instead fix value then we have the flexibility to use the program for multiple times.

}

# Variable

class **Demo**

{

**public static void** **main**(**String** [ ] args)

{

**System.**out**.println** ((“Addition is: “+(15+5));

**System.**out**.println** (“Subtraction is: “+(15-5));

**System.**out**.println** (“Multiplication is: “+(15\*5));

}

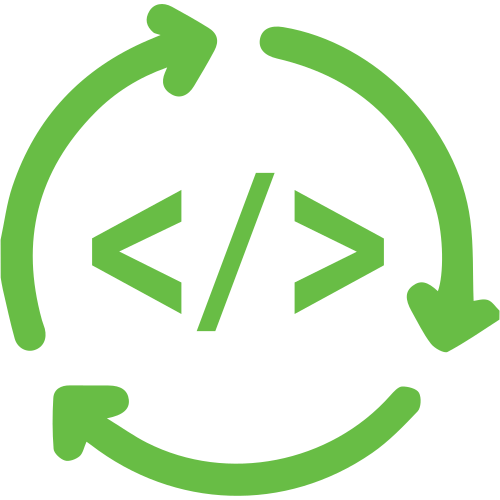
O/p-  **Addition is: 20**

**Subtraction is: 10**

**Multiplication is: 75**

}

## Why we use variable?

* In a worldwide programming field dealing directly with the data/program is not at all recommend approach.
* This is because any kind of data is not a onetime use content.
* To overcame this situation in every programming language variable exist.
* Variable are nothing but a piece of memory which is use store a data.



Info

Why we should go for variable?

1. To avoid hard coding of data.
2. Maintenance of code easier.
3. We can achieve information reusability.

* Before declaring variable programmer need to be design which types of data variable can store.
* It is compulsory to decided which type of data.
* Base on data type our code behave differently.

## Data Types-

* Data types is used to provide instruction to variables regarding what type of data it can store.

There are 2 types of data type

1. Primitive Data type.
2. Non-Primitive Data type.

## Primitive Data-

* The data type which is predefine in java in the form of keyword is called primitive data type.
* All primitive data type are fixed in their size.
* As all primitive data type are keyword, so they are compulsory to declared in lower case.

## Range of Data-

* Formula = No of value we can store is = 2Bit.
* Example = Byte memory is 1byte. So it can store up to 28 no of values that is nothing but 256 nos. Then Range will be (2Bit/2) =(28/2)=(256/2)= (-128 to 127)